Intel® Atom™ Processor Z5xx$^{\Delta}$ Series

Specification Update

Intel® Atom™ Processor Z540$^{\Delta}$, Z530$^{\Delta}$, Z520$^{\Delta}$, Z510$^{\Delta}$, and Z500$^{\Delta}$ on 45 nm process technology

April 2008

Document Number: 319536-001US

Contents

Revision History

Revision	Description	Date
-001	• Initial release	April 2008

§

Preface

This document is an update to the specifications contained in the documents listed in the following Affected Documents/Related Documents table. It is a compilation of device and document errata and specification clarifications and changes, and is intended for hardware system manufacturers and for software developers of applications, operating system, and tools.

Information types defined in the Nomenclature section of this document are consolidated into this update document and are no longer published in other documents. This document may also contain information that has not been previously published.

Affected Documents

Document Title	Document Number/Location
Intel® Atom™ Processor Z5xx Series Datasheet	319535-001US

Related Documents

Document Title	Document Number/Location
Intel® 64 and IA-32 Architectures Software Developer's Manual Documentation Changes	252046
Intel® 64 and IA-32 Architectures Software Developer’s Manual, Volume 1: Basic Architecture	253665
Intel® 64 and IA-32 Architectures Software Developer’s Manual, Volume 2A: Instruction Set Reference, A-M	253666
Intel® 64 and IA-32 Architectures Software Developer’s Manual, Volume 2B: Instruction Set Reference, N-Z	253667
Intel® 64 and IA-32 Architectures Software Developer’s Manual, Volume 3A: System Programming Guide	253668
Intel® 64 and IA-32 Architectures Software Developer’s Manual, Volume 3B: System Programming Guide	253669
IA-32 Intel® Architectures Optimization Reference Manual	248966
Intel® Processor Identification and the CPUID Instruction Application Note (AP-485)	241618
Intel® 64 and IA-32 Architectures Application Note TLBs, Paging-Structure Caches, and Their Invalidation	317080

Nomenclature

Errata are design defects or errors. These may cause the Intel® Atom™ processor Z5xx series on 45-nm process behavior to deviate from published specifications. Hardware and software designed to be used with any given stepping must assume that all errata documented for that stepping are present on all devices.

S-Spec Number is a five-digit code used to identify products. Products are differentiated by their unique characteristics, for example, core speed, L2 cache size, package type, etc. as described in the processor identification information table. Read all notes associated with each S-Spec number.

QDF Number is a four digit code used to distinguish between engineering samples. These samples are used for qualification and early design validation. The functionality of these parts can range from mechanical only to fully functional. This document has a processor identification information table that lists these QDF numbers and the corresponding product details.

Specification Changes are modifications to the current published specifications. These changes will be incorporated in any new release of the specification.

Specification Clarifications describe a specification in greater detail or further highlight a specification's impact to a complex design situation. These clarifications will be incorporated in any new release of the specification.

Documentation Changes include typos, errors, or omissions from the current published specifications. These will be incorporated in any new release of the specification.

Note: Errata remain in the specification update throughout the product's lifecycle, or until a particular stepping is no longer commercially available. Under these circumstances, errata removed from the specification update are archived and available upon request. Specification changes, specification clarifications and documentation changes are removed from the specification update when the appropriate changes are made to the appropriate product specification or user documentation (datasheets, manuals, etc.).

§

Summary Tables of Changes

The following table indicates the Specification Changes, Errata, Specification Clarifications or Documentation Changes, which apply to the listed steppings. Intel intends to fix some of the errata in a future stepping of the component, and to account for the other outstanding issues through documentation or Specification Changes as noted. This table uses the following notations:

Codes Used in Summary Table

Stepping

X:	Erratum, Specification Change or Clarification that applies to this stepping.
(No mark) or (Blank Box):	This erratum is fixed in listed stepping or specification change does not apply to list stepping.

Status

Doc:	Document change or update that will be implemented.
Plan Fix:	This erratum may be fixed in a future stepping of the product.
Fixed:	This erratum has been previously fixed.
No Fix:	There are no plans to fix this erratum.

Row

Shaded:	This item is either new or modified from the previous version of the document.

Note: Each Specification Update item is prefixed with a capital letter to distinguish the product. The key below details the letters that are used in Intel's microprocessor Specification Updates:

A = Dual-Core Intel® Xeon® processor 7000 sequence

C = Intel® Celeron® processor

D = Dual-Core Intel® Xeon® processor 2.80 GHz

E = Intel® Pentium® III processor

F = Intel® Pentium® processor Extreme Edition and Intel® Pentium® D processor

I = Dual-Core Intel® Xeon® processor 5000 series

J = 64-bit Intel® Xeon® processor MP with 1-MB L2 Cache

K = Mobile Intel® Pentium® III processor

L = Intel® Celeron® D processor

M = Mobile Intel® Celeron® processor

N = Intel ® Pentium® 4 processor

O = Intel ® Xeon® processor MP

P = Intel ® Xeon® processor

Q = Mobile Intel® Pentium® 4 processor supporting Hyper-Threading Technology on 90-nm process technology

R = Intel® Pentium® 4 processor on 90 nm process

S = 64-bit Intel® Xeon® processor with 800 MHz system bus (1 MB and 2 MB L2 cache versions)

T = Mobile Intel® Pentium® 4 processor–M

U = 64-bit Intel® Xeon® processor MP with up to 8MB L3 Cache

V = Mobile Intel® Celeron® processor on .13 Micron Process in Micro-FCPGA Package

W= Intel® Celeron®-M processor

X = Intel® Pentium® M processor on 90-nm process with 2-MB L2 cache and Intel® Processors A100 and A110 with 512-KB L2 cache

Y = Intel® Pentium® M processor

Z = Mobile Intel® Pentium® 4 processor with 533 MHz system bus

AA= Intel® Pentium® D Processor 900 Sequence and Intel® Pentium® processor Extreme Edition 955, 965

AB= Intel® Pentium® 4 processor 6x1 Sequence

AC= Intel® Celeron® processor in 478 pin package

AD = Intel® Celeron® D processor on 65 nm process

AE = Intel® Core™ Duo processor and Intel® Core™ Solo processor on 65nm process

AF = Dual-Core™ Intel® Xeon® processor LV

AG = Dual-Core Intel® Xeon® processor 5100 Series

AH= Intel® Core™2 Duo mobile processor

AI = Intel® Core™2 Extreme processor X6800$^{\Delta}$ and Intel® Core™2 Duo Desktop processor E6000 and E4000 Sequence

AJ = Quad-Core Intel® Xeon® processor 5300 Series

AK = Intel® Core™2 Extreme quad-core processor QX6700 and Intel® Core™2 Quad processor Q6600

AL = Dual-Core Intel® Xeon® processor 7100 Series

AN = Intel® Pentium® Dual-Core processor

AO = Quad-Core Intel® Xeon® processor 3200 Series

AP = Dual-Core Intel® Xeon® processor 3000 Series

AQ = Intel® Pentium® Dual-Core Desktop Processor E2000 Sequence

AR = Intel® Celeron® Processor 500 Series

AR = Intel® Celeron processor 500 series

AS = Intel® Xeon® processor 7200, 7300 series

AV = Intel® Core™2 Extreme processor QX9000 sequence and Intel® Core™2 Quad processor Q9000 sequence processor

AW = Intel® Core™ 2 Duo

AX =Quad-Core Intel® Xeon® processor 5400 series

AY =Dual-Core Intel® Xeon® processor 5200 series

AZ =Intel® Core™2 Duo processor and Intel® Core™2 Extreme processor on 45nm process

AAE = Intel® Atom™ processor Z5xx series

Note: Intel processor numbers are not a measure of performance. Processor numbers differentiate features within each processor family, not across different processor families. See http://www.intel.com/products/processor_number for details.

Number	Stepping	PLAN	ERRATA
	C0		
AAE1	X	No Fix	A Write to an APIC Register Sometimes May Appear to Have Not Occurred
AAE2	X	No Fix	An xTPR Update Transaction Cycle, if Enabled, May be Issued to the FSB after the Processor has Issued a Stop-Grant Special Cycle
AAE3	X	No Fix	The Processor May Report a #TS Instead of a #GP Fault
AAE4	X	No Fix	Writing the Local Vector Table (LVT) when an Interrupt is Pending May Cause an Unexpected Interrupt
AAE5	X	No Fix	MOV To/From Debug Registers Causes Debug Exception
AAE6	X	No Fix	Using 2M/4M Pages When A20M# Is Asserted May Result in Incorrect Address Translations
AAE7	X	No Fix	Values for LBR/BTS/BTM will be Incorrect after an Exit from SMM
AAE8	X	No Fix	Incorrect Address Computed For Last Byte of FXSAVE/FXRSTOR Image Leads to Partial Memory Update
AAE9	X	No Fix	A Thermal Interrupt is Not Generated when the Current Temperature is Invalid
AAE10	X	No Fix	Programming the Digital Thermal Sensor (DTS) Threshold May Cause Unexpected Thermal Interrupts
AAE11	X	No Fix	Returning to Real Mode from SMM with EFLAGS.VM Set May Result in Unpredictable System Behavior
AAE12	X	No Fix	Fault on ENTER Instruction May Result in Unexpected Values on Stack Frame
AAE13	X	No Fix	With TF (Trap Flag) Asserted, FP Instruction That Triggers an Unmasked FP Exception May Take Single Step Trap before Retirement of Instruction
AAE14	X	No Fix	An Enabled Debug Breakpoint or Single Step Trap May Be Taken after MOV SS/POP SS Instruction if it is Followed by an Instruction That Signals a Floating Point Exception
AAE15	X	No Fix	Code Segment Limit/Canonical Faults on RSM May be Serviced before Higher Priority Interrupts/Exceptions and May Push the Wrong Address Onto the Stack
AAE16	X	No Fix	BTS(Branch Trace Store) and PEBS(Precise Event Based Sampling) May Update Memory outside the BTS/PEBS Buffer
AAE17	X	No Fix	Single Step Interrupts with Floating Point Exception Pending May Be Mishandled
AAE18	X	No Fix	Unsynchronized Cross-Modifying Code Operations Can Cause Unexpected Instruction Execution Results
AAE19	X	No Fix	IO_SMI Indication in SMRAM State Save Area May be Set Incorrectly
AAE20	X	No Fix	Writes to IA32_DEBUGCTL MSR May Fail when FREEZE_LBRS_ON_PMI is Set
AAE21	X	No Fix	Address Reported by Machine-Check Architecture (MCA) on L2 Cache Errors May be Incorrect

Number	Stepping	PLAN	ERRATA
	C0		
AAE22	X	No Fix	Pending x87 FPU Exceptions (#MF) Following STI May Be Serviced Before Higher Priority Interrupts
AAE23	X	No Fix	Benign Exception after a Double Fault May Not Cause a Triple Fault Shutdown
AAE24	X	No Fix	IA32_MC1_STATUS MSR Bit[60] Does Not Reflect Machine Check Error Reporting Enable Correctly
AAE25	X	No Fix	If Two Logical Processors Use the Same CR3 Value But Configure APIC Virtualization Differently, Either May Operate as if APIC Virtualization Were Disabled
AAE26	X	No Fix	Split Locked Stores or Locked Stores Through Certain Segments May not Trigger the Monitoring Hardware
AAE27	X	No Fix	When BIST is Enabled, Warm Reset Incorrectly Clears IA32_FEATURE_CONTROL MSR and the Last Exception Record MSRs
AAE28	X	No Fix	A VM Exit Due to a Fault While Delivering a Software Interrupt May Save Incorrect Data into the VMCS
AAE29	X	No Fix	CPUID Instruction Returns Incorrect Brand String
AAE30	X	No Fix	A Logical Processor May Incorrectly Clear Thermal Status Log Indicator During Intel Deep Power Down Technology State Transition
AAE31	X	No Fix	The Instruction Cache Does Not Respond to Snoops When All Logical Processors on a Core Are in an Inactive State

Number	SPECIFICATION CHANGES
	There are no specification changes in this revision of the specification update

Number	SPECIFICATION CLARIFICATIONS
1	Clarification of TRANSLATION LOOKASIDE BUFFERS (TLBS) Invalidation

Number	DOCUMENTATION CHANGES
	There are no Documentation Changes in this revision of the specification update.

Identification Information

The Intel® Atom™ processor Z5xx series on 45 nm process stepping can be identified by the following register contents:

Table 1. Component Identification via Programming Interface

Reserved	Extended Family[1]	Extended Model[2]	Reserved	Processor Type[3]	Family Code[4]	Model Number[5]	Stepping ID[6]
31:28	27:20	19:16	15:13	12	11:8	7:4	3:0
	0000000b	0001b		0b	0110b	1100b	XXXXb

When EAX is initialized to a value of 1, the CPUID instruction returns the Extended Family, Extended Model, Type, Family, Model and Stepping value in the EAX register. Note that the EDX processor signature value after reset is equivalent to the processor signature output value in the EAX register.

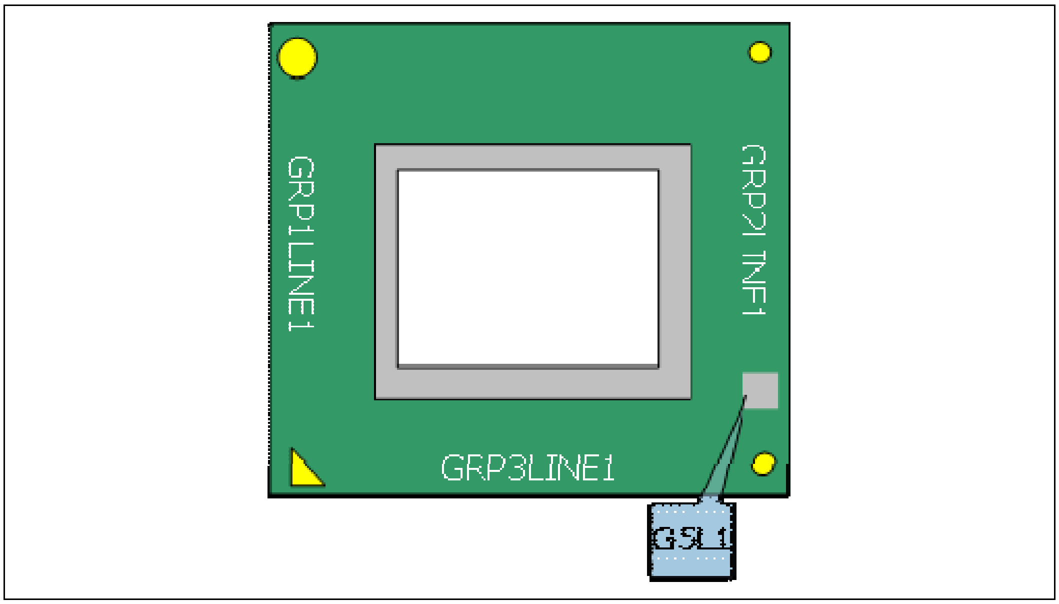
NOTES:

1. The Extended Family, bits [27:20] are used in conjunction with the Family Code, specified in bits [11:8] to indicate whether the processor belongs to the Intel386™, Intel486™, Pentium®, Pentium Pro, Pentium® 4, or Intel Core processor family.
2. The Extended Model, bits [19:16] in conjunction with the Model Number, specified in bits [7:4], are used to identify the model of the processor within the processor's family.
3. The Processor Type, specified in bits [13:12] indicates whether the processor is an original OEM processor, an OverDrive processor, or a dual processor (capable of being used in a dual processor system).
4. The Family Code corresponds to bits [11:8] of the EDX register after RESET, bits [11:8] of the EAX register after the CPUID instruction is executed with a 1 in the EAX register, and the generation field of the Device ID register accessible through Boundary Scan.
5. The Model Number corresponds to bits [7:4] of the EDX register after RESET, bits [7:4] of the EAX register after the CPUID instruction is executed with a 1 in the EAX register, and the model field of the Device ID register accessible through Boundary Scan.
6. The Stepping ID in bits [3:0] indicates the revision number of that model. See Table 2 for the processor stepping ID number in the CPUID information.

Component Marking Information

The Intel® Atom™ processor Z5xx series may be identified by the following component markings.

Figure 1. Intel® Atom™ Processor Z5xx Series (Micro-FCBGA) Markings



GRP1LINE1: INTEL {M}©’07 {e1}
GRP2LINE1: <Level-4 Name><FSB>
GRP3LINE1: {FPO} S-SPEC
G5L1: 2D Matrix

Table 2. Identification Table for Intel® Atom™ Processor Z5xx Series

QDF/ S-Spec	Product Stepping	HFM TDP (W)[1]	Processor Number	FSB Frequency	Processor Signature	Core Speed		Package Micro-FCBGA- Pb = µ-BGA Lead Free
						Highest Freq. Mode (HFM)	Lowest Freq. Mode (LFM)	
SLB6Q	C0	0.65W	Z500	400 MHz	000106C2h	0.8GHz	600 Mhz	FCBGA8
SLB2C	C0	2W	Z510	400 MHz	000106C2h	1.1GHz	600 Mhz	FCBGA8
SLB2H	C0	2W	Z520	533 MHz	000106C2h	1.33GHz	800 Mhz	FCBGA8
SLB6P	C0	2W	Z530	533 MHz	000106C2h	1.60GHz	800 Mhz	FCBGA8
SLB2M	C0	2.4W	Z540	533 MHz	000106C2h	1.86GHz	800 Mhz	FCBGA8

NOTE:

1. Hyper-Threading Technology requires a computer system with an Intel processor supporting Hyper-Threading Technology and an HT Technology enabled chipset, BIOS and operating system. Hyper-threading technology is available on select Intel® Atom™ processor Z5xx series components (Z520=1.33 GHz, Z530=1.60 GHz and Z540=1.86 GHz). HT Technology can add 200 mW of power above TDP, so 1.33 GHz and 1.6GHz processors can have 2.2 W of power and 1.86 GHz processor can have 2.64 W of power when multi-threaded applications are run.

Microcode Updates

Each unique processor stepping/package combination has an associated microcode update that, when applied, constitutes a supported processor (that is, Specified processor = Processor Stepping + Microcode Update). The proper microcode update must be loaded on each processor in a system. The proper microcode update is defined as the latest production microcode update available from Intel for a given family, model and stepping of the processor. Any processor that does not have the correct microcode update loaded is considered to be operating out of specification. Contact your Intel Field Representative to receive the latest production microcode updates.

§

Errata

AAE1 A Write to an APIC Register Sometimes May Appear to Have Not Occurred

Problem: With respect to the retirement of instructions, stores to the uncacheable memory based APIC register space are handled in a non-synchronized way. For example, if an instruction that masks the interrupt flag, for example CLI, is executed soon after an uncacheable write to the Task Priority Register (TPR) that lowers the APIC priority, the interrupt masking operation may take effect before the actual priority has been lowered. This may cause interrupts whose priority is lower than the initial TPR, but higher than the final TPR, to not be serviced until the interrupt enabled flag is finally set (that is, by STI instruction). Interrupts will remain pending and are not lost.

Implication: In this example the processor may allow interrupts to be accepted but may delay their service.

Workaround: This non-synchronization can be avoided by issuing an APIC register read after the APIC register write. This will force the store to the APIC register before any subsequent instructions are executed. No commercial operating system is known to be impacted by this erratum.

Status: For the steppings affected, see the Summary Tables of Changes.

AAE2 An xTPR Update Transaction Cycle, if Enabled, May be Issued to the FSB after the Processor has Issued a Stop-Grant Special Cycle

Problem: According to the FSB (Front Side Bus) protocol specification, no FSB cycles should be issued by the processor once a Stop-Grant special cycle has been issued to the bus. If xTPR update transactions are enabled by clearing the IA32_MISC_ENABLES[bit 23] at the time of Stop-Clock assertion, an xTPR update transaction cycle may be issued to the FSB after the processor has issued a Stop Grant Acknowledge transaction.

Implication: When this erratum occurs in systems using C-states C2 (Stop-Grant State) and higher the result could be a system hang.

Workaround: BIOS must leave the xTPR update transactions disabled (default).

Status: For the steppings affected, see the Summary Tables of Changes.

AAE3 Processor May Report a #TS Instead of a #GP Fault

Problem: A jump to a busy TSS (Task-State Segment) may cause a #TS (invalid TSS exception) instead of a #GP fault (general protection exception).

Implication: Operation systems that access a busy TSS may get invalid TSS fault instead of a #GP fault. Intel has not observed this erratum with any commercially available software.

Workaround: None

Status: For the steppings affected, see the Summary Tables of Changes.

AAE4 Writing the Local Vector Table (LVT) when an Interrupt is Pending May Cause an Unexpected Interrupt

Problem: If a local interrupt is pending when the LVT entry is written, an interrupt may be taken on the new interrupt vector even if the mask bit is set.

Implication: An interrupt may immediately be generated with the new vector when a LVT entry is written, even if the new LVT entry has the mask bit set. If there is no Interrupt Service Routine (ISR) set up for that vector the system will GP fault. If the ISR does not do an End of Interrupt (EOI) the bit for the vector will be left set in the in-service register and mask all interrupts at the same or lower priority.

Workaround: Any vector programmed into an LVT entry must have an ISR associated with it, even if that vector was programmed as masked. This ISR routine must do an EOI to clear any unexpected interrupts that may occur. The ISR associated with the spurious vector does not generate an EOI, therefore the spurious vector should not be used when writing the LVT.

Status: For the steppings affected, see the Summary Tables of Changes.

AAE5 MOV To/From Debug Registers Causes Debug Exception

Problem: When in V86 mode, if a MOV instruction is executed to/from a debug registers, a general-protection exception (#GP) should be generated. However, in the case when the general detect enable flag (GD) bit is set, the observed behavior is that a debug exception (#DB) is generated instead.

Implication: With debug-register protection enabled (that is, the GD bit set), when attempting to execute a MOV on debug registers in V86 mode, a debug exception will be generated instead of the expected general-protection fault.

Workaround: In general, operating systems do not set the GD bit when they are in V86 mode. The GD bit is generally set and used by debuggers. The debug exception handler should check that the exception did not occur in V86 mode before continuing. If the exception did occur in V86 mode, the exception may be directed to the general-protection exception handler.

Status: For the steppings affected, see the Summary Tables of Changes.

AAE6 Using 2M/4M Pages When A20M# Is Asserted May Result in Incorrect Address Translations

Problem: An external A20M# pin if enabled forces address bit 20 to be masked (forced to zero) to emulates real-address mode address wraparound at 1 megabyte. However, if all of the following conditions are met, address bit 20 may not be masked.

- paging is enabled
- a linear address has bit 20 set
- the address references a large page
- A20M# is enabled

Implication: When A20M# is enabled and an address references a large page the resulting translated physical address may be incorrect. This erratum has not been observed with any commercially available operating system.

Workaround: Operating systems should not allow A20M# to be enabled if the masking of address bit 20 could be applied to an address that references a large page. A20M# is normally only used with the first megabyte of memory.

Status: For the steppings affected, see the Summary Tables of Changes.

AAE7 Values for LBR/BTS/BTM will be Incorrect after an Exit from SMM

Problem: After a return from SMM (System Management Mode), the CPU will incorrectly update the LBR (Last Branch Record) and the BTS (Branch Trace Store), hence rendering their data invalid. The corresponding data if sent out as a BTM on the system bus will also be incorrect.
Note: This issue would only occur when one of the 3 above mentioned debug support facilities are used.

Implication: The value of the LBR, BTS, and BTM immediately after an RSM operation should not be used.

Workaround: None

Status: For the steppings affected, see the Summary Tables of Changes.

AAE8 Incorrect Address Computed For Last Byte of FXSAVE/FXRSTOR Image Leads to Partial Memory Update

Problem: A partial memory state save of the 512-byte FXSAVE image or a partial memory state restore of the FXRSTOR image may occur if a memory address exceeds the 64KB limit while the processor is operating in 16-bit mode or if a memory address exceeds the 4GB limit while the processor is operating in 32-bit mode.

Implication: FXSAVE/FXRSTOR will incur a #GP fault due to the memory limit violation as expected but the memory state may be only partially saved or restored.

Workaround: Software should avoid memory accesses that wrap around the respective 16-bit and 32-bit mode memory limits.

Status: For the steppings affected, see the Summary Tables of Changes.

AAE9 A Thermal Interrupt is Not Generated when the Current Temperature is Invalid

Problem: When the DTS (Digital Thermal Sensor) crosses one of its programmed thresholds it generates an interrupt and logs the event (IA32_THERM_STATUS MSR (019Ch) bits [9,7]). Due to this erratum, if the DTS reaches an invalid temperature (as indicated IA32_THERM_STATUS MSR bit[31]) it does not generate an interrupt even if one of the programmed thresholds is crossed and the corresponding log bits become set.

Implication: When the temperature reaches an invalid temperature the CPU does not generate a Thermal interrupt even if a programmed threshold is crossed.

Workaround: None

Status: For the steppings affected, see the Summary Tables of Changes.

AAE10 Programming the Digital Thermal Sensor (DTS) Threshold May Cause Unexpected Thermal Interrupts

Problem: Software can enable DTS thermal interrupts by programming the thermal threshold and setting the respective thermal interrupt enable bit. When programming DTS value, the previous DTS threshold may be crossed. This will generate an unexpected thermal interrupt.

Implication: Software may observe an unexpected thermal interrupt occur after reprogramming the thermal threshold.

Workaround: In the ACPI/OS implement a workaround by temporarily disabling the DTS threshold interrupt before updating the DTS threshold value.

Status: For the steppings affected, see the Summary Tables of Changes.

AAE11 Returning to Real Mode from SMM with EFLAGS.VM Set May Result in Unpredictable System Behavior

Problem: Returning back from SMM mode into real mode while EFLAGS.VM is set in SMRAM may result in unpredictable system behavior.

Implication: If SMM software changes the values of the EFLAGS.VM in SMRAM, it may result in unpredictable system behavior. Intel has not observed this behavior in commercially available software.

Workaround: SMM software should not change the value of EFLAGS.VM in SMRAM.

Status: For the steppings affected, see the Summary Tables of Changes.

AAE12 Fault on ENTER Instruction May Result in Unexpected Values on Stack Frame

Problem: The ENTER instruction is used to create a procedure stack frame. Due to this erratum, if execution of the ENTER instruction results in a fault, the dynamic storage area of the resultant stack frame may contain unexpected values (that is, residual stack data as a result of processing the fault).

Implication: Data in the created stack frame may be altered following a fault on the ENTER instruction. Please refer to "Procedure Calls For Block-Structured Languages" in *Intel® 64 and IA-32 Architectures Software Developer's Manual*, Vol. 1, Basic Architecture, for information on the usage of the ENTER instructions. This erratum is not expected to occur in ring 3. Faults are usually processed in ring 0 and stack switch occurs when transferring to ring 0. Intel has not observed this erratum on any commercially available software.

Workaround: None

Status: For the steppings affected, see the Summary Tables of Changes.

AAE13 With TF (Trap Flag) Asserted, FP Instruction That Triggers an Unmasked FP Exception May Take Single Step Trap before Retirement of Instruction

Problem: If an FP instruction generates an unmasked exception with the EFLAGS.TF=1, it is possible for external events to occur, including a transition to a lower power state. When resuming from the lower power state, it may be possible to take the single step trap before the execution of the original FP instruction completes.

Implication: A Single Step trap will be taken when not expected.

Workaround: None

Status: For the steppings affected, see the Summary Tables of Changes.

AAE14 An Enabled Debug Breakpoint or Single Step Trap May Be Taken After MOV SS/POP SS Instruction if it is Followed by an Instruction That Signals a Floating Point Exception

Problem: A MOV SS/POP SS instruction should inhibit all interrupts including debug breakpoints until after execution of the following instruction. This is intended to allow the sequential execution of MOV SS/POP SS and MOV [r/e]SP, [r/e]BP instructions without having an invalid stack during interrupt handling. However, an enabled debug breakpoint or single step trap may be taken after MOV SS/POP SS if this instruction is followed by an instruction that signals a floating point exception rather than a MOV [r/e]SP, [r/e]BP instruction. This results in a debug exception being signaled on an unexpected instruction boundary since the MOV SS/POP SS and the following instruction should be executed atomically.

Implication: This can result in incorrect signaling of a debug exception and possibly a mismatched Stack Segment and Stack Pointer. If MOV SS/POP SS is not followed by a MOV [r/e]SP, [r/e]BP, there may be a mismatched Stack Segment and Stack Pointer on any exception. Intel has not observed this erratum with any commercially available software, or system.

Workaround: As recommended in the Intel® 64 and IA-32 Architectures Software Developer's Manual, the use of MOV SS/POP SS in conjunction with MOV [r/e]SP and [r/e]BP will avoid the failure since the MOV [r/e]SP and [r/e]BP will not generate a floating point exception. Developers of debug tools should be aware of the potential incorrect debug event signaling created by this erratum.

Status: For the steppings affected, see the Summary Tables of Changes.

AAE15 Code Segment Limit/Canonical Faults on RSM May be Serviced before Higher Priority Interrupts/Exceptions and May Push the Wrong Address Onto the Stack

Problem: Normally, when the processor encounters a Segment Limit or Canonical Fault due to code execution, a #GP (General Protection Exception) fault is generated after all higher priority interrupts and exceptions are serviced. Due to this erratum, if RSM (Resume from System Management Mode) returns to execution flow that results in a Code Segment Limit or Canonical Fault, the #GP fault may be serviced before a higher priority Interrupt or Exception (for example NMI (Non-Maskable Interrupt), Debug break(#DB), Machine Check (#MC), etc.). If the RSM attempts to return to a non-canonical address, the address pushed onto the stack for this #GP fault may not match the non-canonical address that caused the fault.

Implication: Operating systems may observe a #GP fault being serviced before higher priority interrupts and exceptions. Intel has not observed this erratum on any commercially available software.

Workaround: None

Status: For the steppings affected, see the Summary Tables of Changes.

AAE16 BTS(Branch Trace Store) and PEBS(Precise Event Based Sampling) May Update Memory outside the BTS/PEBS Buffer

Problem: If the BTS/PEBS buffer is defined such that:

- The difference between BTS/PEBS buffer base and BTS/PEBS absolute maximum is not an integer multiple of the corresponding record sizes
- BTS/PEBS absolute maximum is less than a record size from the end of the virtual address space
- The record that would cross BTS/PEBS absolute maximum will also continue past the end of the virtual address space

A BTS/PEBS record can be written that will wrap at the 4G boundary (legacy mode) or 2^64 boundary (IA-32e mode, including both 64-bit mode and compatibility mode), and write memory outside of the BTS/PEBS buffer.

Implication: Software that uses BTS/PEBS near the 4G boundary (legacy mode) or 2^64 boundary (IA-32e mode, including both 64-bit mode and compatibility mode), and defines the buffer such that it does not hold an integer multiple of records can update memory outside the BTS/PEBS buffer.

Workaround: Define BTS/PEBS buffer such that BTS/PEBS absolute maximum minus BTS/PEBS buffer base is integer multiple of the corresponding record sizes as recommended in the Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3B.

Status: For the steppings affected, see the Summary Tables of Changes.

AAE17 Single Step Interrupts with Floating Point Exception Pending May Be Mishandled

Problem: In certain circumstances, when a floating point exception (#MF) is pending during single-step execution, processing of the single-step debug exception (#DB) may be mishandled.

Implication: When this erratum occurs, #DB will be incorrectly handled as follows:

- #DB is signaled before the pending higher priority #MF (Interrupt 16)
- #DB is generated twice on the same instruction

Workaround: None

Status: For the steppings affected, see the Summary Tables of Changes.

AAE18 Unsynchronized Cross-Modifying Code Operations Can Cause Unexpected Instruction Execution Results

Problem: The act of one processor, or system bus master, writing data into a currently executing code segment of a second processor with the intent of having the second processor execute that data as code is called cross-modifying code (XMC). XMC that does not force the second processor to execute a synchronizing instruction, prior to execution of the new code, is called unsynchronized XMC. Software using unsynchronized XMC to modify the instruction byte stream of a processor can see unexpected or unpredictable execution behavior from the processor (that is, executing the modified code).

Implication: In this case, the phrase "unexpected or unpredictable execution behavior" encompasses the generation of most of the exceptions listed in the Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3A: System Programming Guide, including a General Protection Fault (#GP) or other unexpected behaviors.

Workaround: In order to avoid this erratum, programmers should use the XMC synchronization algorithm as detailed in the *Intel® 64 and IA-32 Architectures Software Developer's Manual*, *Volume 3A: System Programming Guide*, Section: Handling Self- and Cross-Modifying Code.

Status: For the steppings affected, see the Summary Tables of Changes.

AAE19 IO_SMI Indication in SMRAM State Save Area May be Set Incorrectly

Problem: The IO_SMI bit in SMRAM's location 7FA4H is set to "1" by the CPU to indicate a System Management Interrupt (SMI) occurred as the result of executing an instruction that reads from an I/O port. Due to this erratum, the IO_SMI bit may be incorrectly set by:

- A SMI that is, pending while a lower priority event is executing
- A REP I/O read
- A I/O read that redirects to MWAIT

Implication: SMM handlers may get false IO_SMI indication.

Workaround: The SMM handler has to evaluate the saved context to determine if the SMI was triggered by an instruction that read from an I/O port. The SMM handler must not restart an I/O instruction if the platform has not been configured to generate a synchronous SMI for the recorded I/O port address.

Status: For the steppings affected, see the Summary Tables of Changes.

AAE20 Writes to IA32_DEBUGCTL MSR May Fail when FREEZE_LBRS_ON_PMI is Set

Problem: When the FREEZE_LBRS_ON_PMI, IA32_DEBUGCTL MSR (1D9H) bit [11], is set, future writes to IA32_DEBUGCTL MSR may not occur in certain rare corner cases. Writes to this register by software or during certain processor operations are affected.

Implication: Under certain circumstances, the IA32_DEBUGCTL MSR value may not be updated properly and will retain the old value. Intel has not observed this erratum with any commercially available software.

Workaround: Do not set the FREEZE_LBRS_ON_PMI bit of IA32_DEBUGCTL MSR.

Status: For the steppings affected, see the Summary Tables of Changes.

AAE21 Address Reported by Machine-Check Architecture (MCA) on L2 Cache Errors May be Incorrect

Problem: When an L2 Cache error occurs (Error code 0x010A or 0x110A reported in IA32_MCi_STATUS MSR bits [15:0]), the address is logged in the MCA address register (IA32_MCi_ADDR MSR). Under some scenarios, the address reported may be incorrect.

Implication: Software should not rely on the value reported in IA32_MCi_ADDR MSR for L2 Cache errors.

Workaround: None

Status: For the steppings affected, see the Summary Tables of Changes.

AAE22 Pending x87 FPU Exceptions (#MF) Following STI May Be Serviced Before Higher Priority Interrupts

Problem: Interrupts that are pending prior to the execution of the STI (Set Interrupt Flag) instruction are normally serviced immediately after the instruction following the STI. An exception to this is if the following instruction triggers a #MF. In this situation, the interrupt should be serviced before the #MF. Because of this erratum, if following STI, an instruction that triggers a #MF is executed while STPCLK#, Enhanced Intel SpeedStep Technology transitions or Thermal Monitor events occur, the pending #MF may be serviced before higher priority interrupts.

Implication: Software may observe #MF being serviced before higher priority interrupts.

Workaround: None

Status: For the steppings affected, see the Summary Tables of Changes.

AAE23 Benign Exception after a Double Fault May Not Cause a Triple Fault Shutdown

Problem: According to the Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3A, "Exception and Interrupt Reference", if another exception occurs while attempting to call the double-fault handler, the processor enters shutdown mode. Due to this erratum, any benign faults while attempting to call double-fault handler will not cause a shutdown. However Contributory Exceptions and Page Faults will continue to cause a triple fault shutdown.

Implication: If a benign exception occurs while attempting to call the double-fault handler, the processor may hang or may handle the benign exception. Intel has not observed this erratum with any commercially available software.

Workaround: None

Status: For the steppings affected, see the Summary Tables of Changes.

AAE24 IA32_MC1_STATUS MSR Bit [60] Does Not Reflect Machine Check Error Reporting Enable Correctly

Problem: IA32_MC1_STATUS MSR (405H) bit[60] (EN- Error Enabled) is supposed to indicate whether the enable bit in the IA32_MC1_CTL MSR (404H) was set at the time of the last update to the IA32_MC1_STATUS MSR. Due to this erratum, IA32_MC1_STATUS MSR bit [60] instead reports the current value of the IA32_MC1_CTL MSR enable bit.

Implication: IA32_MC1_STATUS MSR bit [60] may not reflect the correct state of the enable bit in the IA32_MC1_CTL MSR at the time of the last update.

Workaround: None

Status: For the steppings affected, see the Summary Tables of Changes.

AAE25 If Two Logical Processors Use the Same CR3 Value But Configure APIC Virtualization Differently, Either May Operate as if APIC Virtualization Were Disabled

Problem: If a logical processor is in VMX non-root operation with the "virtual APIC accesses" VM-execution control set to 1, it may incorrectly operate as if the "virtual APIC accesses" VM-execution control was cleared to 0 if another logical processor has the same value in CR3 and one of the following is true:

- The other logical processor is not in VMX non-root operation
- The other logical processor has the “virtualize APIC accesses” VM-execution control cleared to 0
- The other logical processor’s value of the “APIC-access address” VM-execution control field is different than that of the first logical processor

Implication: A logical processor may fail to support the APIC-virtualization features properly if a virtual-machine monitor (VMM) uses the same page tables as a virtual machine (VM) using the APIC-virtualization features, or if two VMs (or two virtual CPUs within a VM) use the same page tables but operate with different settings of the APIC-virtualization features.

Workaround: A VMM should not use for itself the same page tables as a VM using the APIC-virtualization features, and it should configure two virtual CPUs to use the same page tables only if they use the same settings of the APIC-virtualization features.

Status: For the steppings affected, see the Summary Tables of Changes.

AAE26 Split Locked Stores or Locked Stores Through Certain Segments May not Trigger the Monitoring Hardware

Problem: Logical processors normally resume program execution following the MWAIT, when another logical processor performs a write access to a WB cacheable address within the address range used to perform the MONITOR operation. Due to this erratum, a logical processor may not resume execution until the next targeted interrupt event or O/S timer tick following a locked store within the monitored address range that either spans across cache lines or uses a segment register whose segment base is non-cacheline aligned.

Implication: The logical processor that executed the MWAIT instruction may not resume execution until the next targeted interrupt event or O/S timer tick in the case where the monitored address is written by a locked store which is either split across cache lines or through a segment whose segment base bits 5 to 0 are non-zero.

Workaround: Avoid accessing the monitored address range using either locked stores that split cache lines or locked stores that use a segment with a non-cacheline aligned segment base. It is possible for the BIOS to contain a workaround for this erratum

Status: For the steppings affected, see the Summary Tables of Changes.

AAE27 When BIST is Enabled, Warm Reset Incorrectly Clears IA32_FEATURE_CONTROL MSR and the Last Exception Record MSRs

Problem: IA32_FEATURE_CONTROL MSR (3AH), MSR_LER_FROM_LIP MSR (1DDH), and MSR_LER_TO_LIP MSR (1DEH) are cleared during warm reset when BIST (Built-In Self Test) is enabled. These MSRs should only be cleared on a power-up reset and not on a warm reset. A warm reset is different from a power-up reset in that PWRGOOD remains active throughout the assertion of RESET#.

Implication: Due to this erratum, any warm reset will clear IA32_FEATURE_CONTROL MSR, MSR_LER_FROM_LIP MSR, and MSR_LER_TO_LIP MSR content when BIST is enabled.

Workaround: BIOS or other firmware software must save IA32_FEATURE_CONTROL MSR, MSR_LER_FROM_LIP MSR, and MSR_LER_TO_LIP MSR information before warm reset and restore and reprogram the MSRs after the warm reset.

Status: For the steppings affected, see the Summary Tables of Changes.

AAE28 A VM Exit Due to a Fault While Delivering a Software Interrupt May Save Incorrect Data into the VMCS

Problem: If a fault occurs during delivery of a software interrupt (INT*n*) in virtual-8086 mode when virtual mode extensions are in effect and that fault causes a VM exit, incorrect data may be saved into the VMCS. Specifically, information about the software interrupt may not be reported in the IDT-vectoring information field. In addition, the interruptibility-state field may indicate blocking by STI or by MOV SS if such blocking were in effect before execution of the INT*n* instruction or before execution of the VM-entry instruction that injected the software interrupt.

Implication: In general, VMM software that follows the guidelines given in the section “Handling VM Exits Due to Exceptions” of *Intel® 64 and IA-32 Architectures Software Developer’s Manual Volume 3B: System Programming Guide* should not be affected. If the erratum improperly causes indication of blocking by STI or by MOV SS, the ability of a VMM to inject an interrupt may be delayed by one instruction.

Workaround: VMM software should follow the guidelines given in the section “Handling VM Exits Due to Exceptions” of *Intel® 64 and IA-32 Architectures Software Developer’s Manual Volume 3B: System Programming Guide*

Status: For the steppings affected, see the Summary Tables of Changes.

AAE29 CPUID Instruction Returns Incorrect Brand String

Problem: When a CPUID instruction is executed with EAX = 80000002H, 80000003H and 80000004H on an Intel® Atom™ processor, the return value contains the brand string *Intel(R) Core(TM) CPU* when it should have *Intel(R) Atom(TM) CPU*.

Implication: When this erratum occurs, the processor will report the incorrect brand string.

Workaround: It is possible for the BIOS to contain a workaround for this erratum.

Status: For the steppings affected, see the Summary Tables of Changes.

AAE30 A Logical Processor May Incorrectly Clear Thermal Status Log Indicator During Intel Deep Power Down Technology State Transition

Problem: When a logical processor enters the Intel Deep Power Down Technology State (for example as requested by MWAIT or I/O redirection), it may incorrectly clear the sticky Thermal Status Log flag (bit 1) in IA32_THERM_STATUS MSR (19CH). This erratum will not occur when Hyper-Threading (HT) is disabled.

Implication: When Hyper-Threading is enabled, a logical processor may incorrectly indicate that the thermal sensor has not tripped since last power-up.

Workaround: It is possible for the BIOS to contain a workaround for this erratum.

Status: For the steppings affected, see the Summary Tables of Changes.

AAE31 The Instruction Cache Does Not Respond to Snoops When All Logical Processors on a Core Are in an Inactive State

Problem: When all logical processors on a core enter an inactive state (for example MWAIT or HLT), the processor may incorrectly stop flushing lines in its instruction cache in response to snoops. This may cause the processor to not detect that memory has been modified and to execute the old instructions after waking up instead of the new contents of memory.

Implication: The processor may execute incorrect instructions after waking up from an inactive state.

Workaround: It is possible for the BIOS to contain a workaround for this erratum

Status: For the steppings affected, see the Summary Tables of Changes.

§

Specification Changes

There are no specification changes in this revision of the specification update.

§

Specification Clarifications

1. Clarification of TRANSLATION LOOKASIDE BUFFERS (TLBS) Invalidation

Section 10.9 INVALIDATING THE TRANSLATION LOOKASIDE BUFFERS (TLBS) of the Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3A: System Programming Guide will be modified to include the presence of page table structure caches, such as the page directory cache, which Intel processors implement. This information is needed to aid operating systems in managing page table structure invalidations properly.

Intel will update the Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3A: System Programming Guide in the coming months. Until that time, an application note, *TLBs, Paging-Structure Caches, and Their Invalidation* (http://www.intel.com/products/processor/manuals/index.htm), is available which provides more information on the paging structure caches and TLB invalidation.

In rare instances, improper TLB invalidation may result in unpredictable system behavior, such as system hangs or incorrect data. Developers of operating systems should take this documentation into account when designing TLB invalidation algorithms.

§

Documentation Changes

There are no Documentation changes in this revision of the specification update.